2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636385

FIDNet: LiDAR Point Cloud Semantic Segmentation with Fully Interpolation Decoding

Yiming Zhao, Lin Bai and Xinming Huang

***Abstract*— Projecting the point cloud on the 2D spherical range image transforms the LiDAR semantic segmentation to a 2D segmentation task on the range image. However, the LiDAR range image is still naturally different from the regular 2D RGB image; for example, each position on the range image encodes the unique geometry information. In this paper, we propose a new projection-based LiDAR semantic segmentation pipeline that consists of a novel network structure and an efficient post-processing step. In our network structure, we design a FID (fully interpolation decoding) module that directly upsamples the multi-resolution feature maps using bilinear interpolation. Inspired by the 3D distance interpolation used in PointNet++, we argue this FID module is a 2D version distance interpolation on (θ, ϕ) space. As a parameter-free decoding module, the FID largely reduces the model complexity by maintaining good performance. Besides the network structure, we empirically find that our model predictions have clear boundaries between different semantic classes. This makes us rethink whether the widely used K-nearest-neighbor post-processing is still necessary for our pipeline. Then, we realize the many-to-one mapping causes the blurring effect that some points are mapped into the same pixel and share the same label. Therefore, we propose to process those occluded points by assigning the nearest predicted label to them. This NLA (nearest label assignment) post-processing step shows a better performance than KNN with faster inference speed in the ablation study. On SemanticKITTI dataset, our pipeline achieves the best performance among all projection-based methods with 64×2048 resolution and all point-wise solutions. With a ResNet-34 as the backbone, both the training and testing of our model can be finished on a single RTX 2080 Ti with 11G memory. The code is released here.[1]**

I. INTRODUCTION

LiDAR sensor is playing an important role in outdoor robots, especially autonomous cars. With the booming of deep learning techniques, recent research topics focus on extracting object and semantic information from the point cloud. LiDAR semantic segmentation is such a task that a neural network predicts the semantic label of each point [1]. As shown in Fig. 1, solving this task builds the 3D understanding of the nearby environment.

Though the first dataset specifically aimed at benchmarking the LiDAR semantic segmentation was published in 2019 [2], the interest in processing 3D point clouds with neural networks arose earlier in two communities. Researchers from the computer vision society investigated how to design a permutation invariant network to deal with more general unordered point clouds [3], [4]. In contrast, the solution design from researchers on the robotic side considered more about the LiDAR sensing mechanism by projecting the point cloud on the 2D spherical range image [5]. The projection-based solution allows the well-designed 2D image semantic segmentation models to be directly used for the LiDAR semantic segmentation task. It lacks the ability to process more general unordered point clouds, but it shows practical advantages such as better performance in terms of both the speed and accuracy [6], [7]. To chase a better performance, recent researchers further design models by combining multi-view projections or voxelization with point-wise features [8], [9], [10].

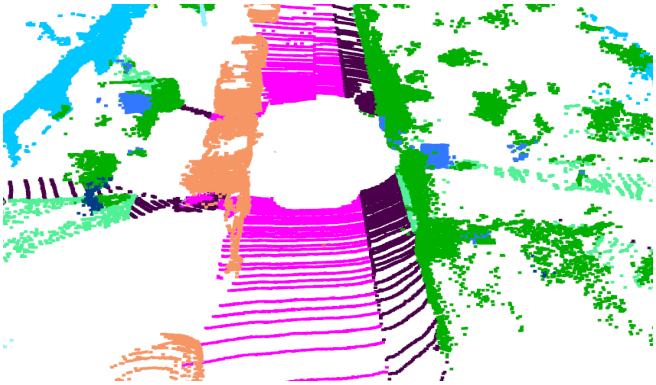

Fig. 1. The LiDAR point cloud semantic segmentation predicts a semantic label for each point to help the car to understand the 3D surroundings. This is a sample from the validation sequence in the SemanticKITTI dataset.

Motivation The major goal of this paper is to argue how should we design the network for projection-based LiDAR segmentation. We hope to keep the structure as common as possible as well as maintain a good performance. Most image-based network structures rely on an encoder-decoder structure. This paper rethinks this structure by considering the difference between the spherical representation image and the regular image. On the spherical representation image, each position is a 3D point with its unique location information. However, on the regular image, two different pixels may have the same spectral information that makes only a pattern of pixels meaningful. To design a better structure for range image, we propose several modules, including an input module with 1×1 convolution, a backbone network to extract multi-scale features, a fully interpolation decoding module, and a final post-processing step that only assigning labels for those occluded points.

Contribution This paper proposes a new pipeline to solve the LiDAR semantic segmentation in a projection fashion. The whole solution is clean and effective. Both the training and testing can be conducted on a single RTX 2080 Ti with

Authors are with Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Massachusetts 01609, USA. yzhao7@wpi.edu

[1]https://github.com/placeforyiming/IROS21-FIDNet-SemanticKITTI

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 04:56:55 UTC from IEEE Xplore. Restrictions apply.

11 G memory. The solution's performance is better than all projection-based methods by following the same 64×2048 input resolution. There are two major technical contributions that we think will be helpful for other related methods:

- *We propose a parameter-free fully interpolation decoding module that only contains bilinear interpolation operation.* Most existing decoder structures need transpose convolution or special combined convolution with interpolation to upsample low-resolution feature maps. We demonstrate that only using bilinear interpolation can already achieve state-of-the-art performance. This setting reduces the model complexity by avoiding a large amount parameters in the decoder used by other models [11], [7].
- *We replace the widely used K-nearest-neighbor post-processing with a more efficient and intuitive step.* K-nearest-neighbor was proposed to solve the boundary-blurring effect generated by two reasons [1], which are the ambiguous output of CNNs and the many-to-one mapping on spherical range image. However, we empirically find the blurring effect on our predictions is negligible. Then, we specifically focus on the many-to-one mapping that some points will be mapped on the same location with others, thus not having directly predicted labels from the network. To solve this problem, we assign the predicted label of the nearest point in 3D space to those unpredicted points. Compared with KNN, this NLA (nearest label assignment) post-processing step has better performance and faster inference speed.

Both of the above two contributions bring practical improvements to the network design of projection-based LiDAR semantic segmentation task, which have not been discussed by other literature as far as we know. There are also some other minor contributions such as an input module with 1×1 convolution or nearby point feature aggregation with atrous convolution [12]. Since similar ideas have been explored in recent papers [13], we will introduce them in the method part, but will not claim them as the major contribution of this paper.

II. RELATED WORK

A. Point-based Networks

PointNets PointNet [3] summarizes several key properties of the general point cloud, including unordered, invariance under transformations, and interaction among points. Those unique challenges stimulate them to design a MLP (multi-layer perceptron) based network structure and a per-point feature vector concatenated by global and point-wise features. Besides only the concatenation of global and point-wise features, PointNet++ [4] propose novel set learning layers to adaptively combine features from multiple scales. However, the local query and grouping limit the model performance on a large point cloud. This attracts some successive papers [14], [6], the best pure point-based solution still lags behind on LiDAR point cloud semantic segmentation [6].

PointConvs Along with the network structure designing, solving the point cloud segmentation with special convolutional kernels is also a hot topic. Some ideas, such as PointCNN [15] and KPConv [16], have been tried on various datasets and show a strong generality. Some of those special convolutional kernels have been merged as part of other solutions [17].

B. Voxel-based Networks

Split the 3D world in discretized voxels is a straightforward idea to process the point cloud. After the voxelization, the regular 3D convolution is able to be used on 3D space just like 2D convolution on 2D image [18], [19]. However, the improvement of those methods in the outdoor LiDAR point cloud remains limited. Some recent papers are trying to consider a better 3D voxelization method to cut the 3D space by incorporating how LiDAR sensor generate the point cloud [20].

C. Projection-based Network

Single-View projection The success of 2D convolution neural networks on the image raises the question of whether it is possible to project the 3D point cloud on 2D space for the sake of using well-developed existing models. The scanning mechanism of the LiDAR sensor suggests the spherical range projection [1]. This idea has been explored by many recent papers from different aspects [7], [21], [13]. Besides the range view, bird-eye-view is also considered by recent papers [9]. All those single view projection methods have the benefit of using 2D convolution networks, like the controllable inference speed.

Multi-View projection With all those newly developed methods above, it is natural to start thinking about how to combine part of different ideas together. In recent multi-view projection solutions [22], [23], MLP based feature extractor is used first, then feature tensors from different views are fused together to be processed by one decoder. However, those methods need to take the extra cost to prepare multi-view projection, and it is also not clear how much improvement compared with single view projection.

D. Practical Considerations

As a new and important challenge, how to design models to solve the LiDAR semantic segmentation is still an open problem. The solution should consider both the numerical indicator and the practical feasibility. Single view projection methods are able to directly use 2D convolutional neural networks, thus do not need to worry about feasibility as there are a bunch of techniques to optimize 2D networks for various application requirements [24], [25]. This paper specifically works on providing a pipeline for spherical range single view projection solutions with better performance.

III. METHOD

The range view projection methods map each point from (x, y, z) Cartesian coordinate to (r, θ, ϕ) spherical coordinate. After discretizing the 2D (θ, ϕ) space, each point will

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 04:56:55 UTC from IEEE Xplore. Restrictions apply.

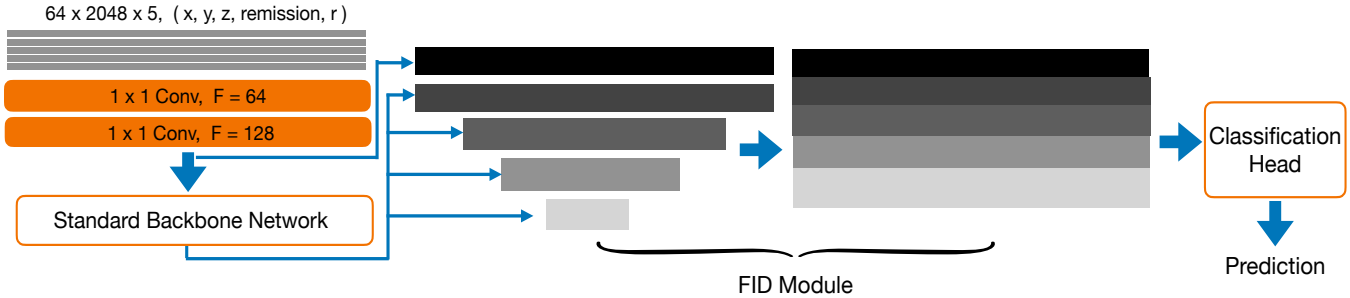

Fig. 2. Illustration of our network structure. The input module has two 1×1 layers mapping each point to a high dimensional space. The backbone can be any regular standard network, like ResNet-34 used in this paper. The FID module upsamples all low-resolution feature maps to the original size and concatenates them together. The last classification head takes in the merged large tensor and outputs the label of each point.

have a mapped position on the (θ, ϕ) image. The same as RGB three channels on regular images, the LiDAR point cloud range image has five channels $(x, y, z, r, remission)$.

A. Input and Backbone Modules

Though the input is like a 2D image with five channels, each position still represents the information of a point. Therefore, we process the input with two 1×1 convolutional layers that map each point to a high dimension tensor. This input processing module is analogous to the PointNet [3] which extracts point features by using MLP (multi-layer perceptron). Then, we feed the high dimensional tensor consists of point-wise features into a regular backbone network. The backbone network can be any structures, such as faster MobileNet [26] or more accurate HRNet [27]. All those CNN structures take a tensor in and generate multi-resolution feature maps. In this paper, we use the standard ResNet-34 [28] as the backbone.

The question is, why the regular backbone network designed for images still can work on point-wise feature tensors? We argue that the LiDAR point cloud is ordered in spherical coordinate as the laser scanner emits laser beams line by line. This mechanism makes it possible to build the special 2D spherical range representation of the LiDAR point cloud. The regular 3×3 convolutional operator in modern backbone networks is equivalent to the eight nearest queries and grouping on the 2D (θ, ϕ) space. Thus, using the backbone network designed for the image to process point-wise features is the 2D correspondence of the 3D K-nearest query and grouping for unordered point cloud used in PointNet++ [4].

B. Fully Interpolation Decoding

In our network structure shown in Fig. 2, we use 1×1 convolution to map each point vector to a high dimension tensor, then extract multi-scale point features with a backbone network that keeps processing nearby point features on 2D (θ, ϕ) space. The next question is, how to fuse that information together in a point-wise way. In PointNet [3], the per-point feature vector is concatenated with the global feature vector as the final feature vector for each point. In PointNet++ [4], the distance-based interpolation is used to upsample a subset of points to a larger set. Those two unique operations inspire us to design a similar module aiming to fuse the multi-scale information.

The distance interpolation defined in PointNet++ [4] is:

$$f(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i}{\sum_{i=1}^{k} w_i(x)}, \quad where \;\; w_i(x) = \frac{1}{d(x, x_i)}.$$

However, on the 2D (θ, ϕ) space, if we set number of nearest points $k = 4$ and define the distance function $d(x, x_i)$ as l_1 distance, the distance interpolation will exactly degenerate to bilinear upsample. Using the bilinear upsample to interpolate the low-resolution feature maps gives us five point-wise feature tensors which have the same resolution but encode different level information. The same as PointNet [3], we concatenate all those feature tensors together. Then, after going through the network structure in Fig. 2, each point is mapped from $(x, y, z, r, remission)$ to a high dimensional vector with multi-level information.

We name this module FID (fully interpolation decoding). Compared with regular decoder used in other networks, FID is completely parameter-free that largely reduces the model complexity and memory cost. This advantage also brings benefits to the network structure design. For example, one can only focus on customizing the backbone to meet the requirement of the hardware limitation.

C. Classification Head

After the FID (fully interpolation decoding) module, we get a per-point feature tensor. To help each point vector better fuse the nearby information, we further use atrous convolution to extract local features and aggregate them by simple concatenation with the original feature tensor. This operation is similar to ASPP (atrous spatial pyramid pooling) [29], the difference is we do not apply average pooling but concatenate the outputs with original feature tensor together. The final feature tensor is sent into 1×1 classification layers to predict the semantic label of each point. This classification head is shown in Fig. 3.

D. Post-processing with Nearest Label Assignment

Almost all recent developed projection-based LiDAR semantic segmentation solutions adopt a KNN (K-nearest-neighbor) post-processing module to alleviate the boundary-blurring effect. As claimed in a recent paper [1], two reasons

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 04:56:55 UTC from IEEE Xplore. Restrictions apply.

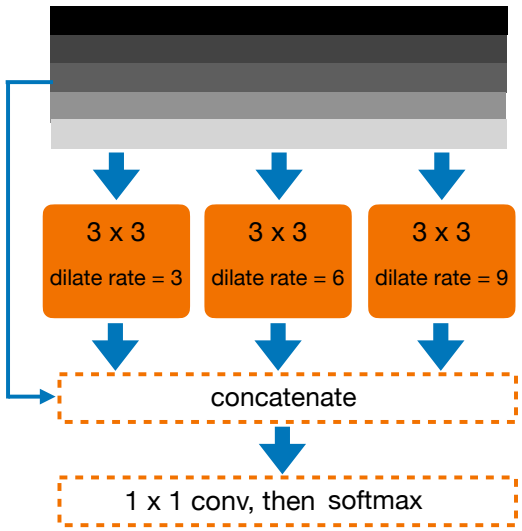

Fig. 3. Illustration of our classification head. This head aggregates the original tensor from the FID (fully interpolation decoding) module with tensors from atrous convolution layers. The final point-wise feature tensor is processed by regular 1×1 convolution layers to get the semantic label.

make this effect happen: blurring outputs from the neural network and the many-to-one mapping on the range image. However, after visualizing some of our network outputs, we realize the FID (fully interpolation decoding) module is already able to give predictions with clear object boundaries. In Fig. 4, we show our observations. At the top, we visualize two examples by only displaying those points that are processed by the network. We can see there is almost no blurring effect even for small poles. At the bottom of Fig. 4, it is clear to see adding those occluded points creates blurring boundaries. This stimulates us to rethink if it is still necessary to keep the KNN post-processing step in our pipeline.

Here we firstly give more discussions about the many-to-one mapping problem. The mapping from (x, y, z) Cartesian coordinate to (r, θ, ϕ) spherical coordinate is a one-to-one continuous mapping. However, discretizing (r, θ, ϕ) on 2D (θ, ϕ) image will group some close points in one cell. After projection, only the information of one point of the cell will be processed by the neural network. Note, those points mapped on the same cell are only close on 2D (θ, ϕ) space, and they may have a large distance with the processed one. The potential large distance indicates those occluded points may have different labels with the predicted point.

Based on the phenomenon in Fig. 4, we argue the many-to-one mapping is the major issue in our pipeline. From the discussion above, we know those points mapped on the same cell may have large distances with each other, thus do not share the same label. This implies a simple solution that we can directly assign the label of the nearest point in 3D space to those occluded points. Similar KNN, we design our NLA (nearest label assignment), in Alg. 1, on range image to find the nearest point in a local patch that is GPU enabled.

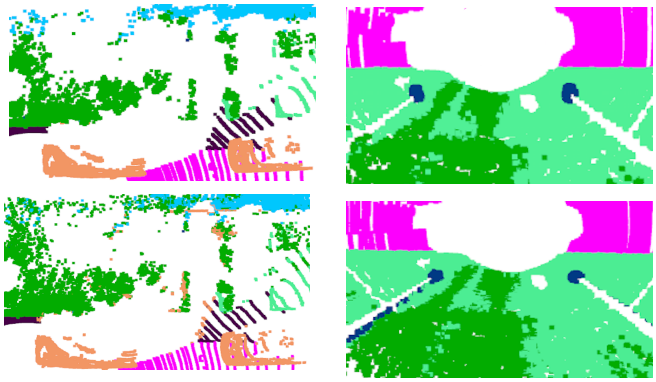

Fig. 4. At the **(top)**, we visualize those points that are mapped on the range image. Those labels are directly predicted from the neural network and have clear boundaries. At the **(bottom)**, we add the other points that are mapped on already occupied pixels (more than one point mapped on one pixel). Some of those points are from object boundaries, thus do not share the same label as the predicted one on the pixel. Directly assigning predicted labels will cause the blurred boundary effect as Middle-Red points around the car on the bottom-left image and the Dark-Blue points around poles on the bottom-right image.

Algorithm 1 Nearest Label Assignment (NLA)

Input : Range image I_r with size $H \times W$,
predicted label map I_{label} with size $H \times W$,
vector R_{all} with range values for all points,
vector h_{all} with projected h values for all points,
vector w_{all} with projected w values for all points,
local kernel size k.

Output: Vector $Labels$ with predicted labels for all points.

```
Labels ← empty list [ ],  k ← 5
S(h, w, k) ← ∀(h_n, w_m), where (h_n, w_m) in the k × k local patch centered at (h, w);
foreach i in 1 : R_all.length() do
    min_diff ← +∞;
    foreach position (h_n, w_m) in S(h_all[i], w_all[i], k) do
        if abs(I_r(h_n, w_m) − R_all[i]) < min_diff then
            label_each = I_label(h_n, w_m)
            min_diff = abs(I_r(h_n, w_m) − R_all[i])
        end
        Labels.append(label_each)
    end
end
return Labels
```

Compared with KNN, our NLA post-processing step does not need Gaussian weighting and range cutoff, thus is a less complex solution. In the ablation study, we show our nearest label assignment post-processing step has better mIoU with faster inference speed than KNN.

E. Other Training Settings

For the data augmentation, we followed other papers to do the rotation and flipping along the y axis [23], [7]. We set the batch size as 2 and adopted the Adam optimizer with a one-cycle learning rate policy. The maximum learning rate was set to 0.002, and the total training epoch was set to 30. For the loss function, we combined the weighted cross-entropy

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 04:56:55 UTC from IEEE Xplore. Restrictions apply.

TABLE I

THE PERFORMANCE COMPARISON ON SEMANTICKITTI TEST SET (SEQUENCE 11 TO SEQUENCE 21). THE UPPER-HALF ARE POINT-WISE METHODS, AND THE LOWER-HALF ARE PROJECTION-BASED METHODS.

Methods	Size	**mean-IoU**	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
PointNet [3]	50K pts	14.6	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7
PointNet++ [4]	50 K pts	20.1	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9
SPGraph [14]	50K pts	20.0	68.3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5
SPLATNet [30]	50K pts	22.8	66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0
TangentConv [31]	50K pts	35.9	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2
PointASNL [32]	8K pts	46.8	87.9	0.0	25.1	39.0	29.2	34.2	57.6	0.0	87.4	24.3	74.3	1.8	83.1	43.9	84.1	52.2	**70.6**	57.8	36.9
LatticeNet [33]	50 K pts	52.9	92.9	16.6	22.2	26.6	21.4	35.6	43.0	**46.0**	90.0	59.4	74.1	22.0	88.2	58.8	81.7	63.6	63.1	51.9	48.4
RandLa-Net [6]	50K pts	53.9	94.2	26.0	25.8	**40.1**	38.9	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7
S-BKI [34]	all	51.3	83.8	30.6	43.0	26.0	19.6	8.5	3.4	0.0	**92.6**	**65.3**	**77.4**	30.1	89.7	63.7	83.4	64.3	67.4	**58.6**	**67.1**
KPConv [16]	50K pts	58.8	**96.0**	30.2	42.5	33.4	**44.3**	61.5	**61.6**	11.8	88.8	61.3	72.7	**31.6**	**90.5**	64.2	**84.8**	**69.2**	69.1	56.4	47.4
SqueezeSeg-CRF [5]	64×2048	30.8	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3
SqueezeSegV2-CRF [35]	64×2048	39.6	82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0
SqueezeSegV3 [21]	64×2048	55.9	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9
RangeNet53++KNN [1]	64×2048	52.2	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
SalsaNet [36]	64×2048	45.4	87.5	26.2	24.6	24.0	17.5	33.2	31.1	8.4	89.7	51.7	70.7	19.7	82.8	48.0	73.0	40.0	61.7	31.3	41.9
SalsaNext [7]	64×2048	**59.5**	91.9	48.3	38.6	38.9	31.9	60.2	59.0	19.4	91.7	63.7	75.8	29.1	90.2	**64.2**	81.8	63.6	66.5	54.3	62.1
PolarNet [37]	[480, 360, 32]	54.3	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	90.0	61.3	84.0	65.5	67.8	51.8	57.5
3D-MiniNet-KNN [13]	64×2048	55.8	90.5	42.3	42.1	28.5	29.4	47.8	44.1	14.5	91.6	64.2	74.5	25.4	89.4	60.8	82.8	60.8	66.7	48.0	56.6
Ours	64×2048	**59.5**	93.9	**54.7**	**48.9**	27.6	23.9	**62.3**	59.8	23.7	90.6	59.1	75.8	26.7	88.9	60.5	84.5	64.4	69.0	53.3	62.8

loss [38] and the Lovász-Softmax loss [39] together. Thanks to the parameter-free FID (fully interpolation decoding) module, all our experiments were conducted on a single RTX 2080 Ti with the mix-precision choice in PyTorch.

IV. EXPERIMENT

A. Dataset

The SemanticKITTI dataset [2] is a recent large-scale dataset that provides dense point-wise annotations for the entire KITTI Odometry Benchmark [40]. The dataset consists of 22 sequences in total. In this paper, we strictly follow the official split to train the model on sequences 00 to 07, and sequences 09, 10. Sequence 08 is used as the validation set to help us choose the best checkpoint. We submit our predictions on sequences 11 to 21 and report the result from the leaderboard to compare with others.

B. Performance Comparison

We compare our model performance with other methods in Table I. As we design the structure to process the projected range image by considering point properties, we mainly compare our method with the point-wise solutions as well as the projection-based solutions. By following the same settings, our pipeline outperforms all point-wise solutions with a large margin and achieves similar performance with the best projection-based solutions.

C. Ablation Study

In this paper, we empirically find our network predictions do not have a strong edge blurring effect. The blurring on the boundary is mainly caused by the many-to-one mapping problem. To solve this, we design a new post-processing algorithm that simply assigns the label of the nearest point in 3D space to those points without predictions from the neural network. In Table II, we can see this NLA (Nearest Label Assignment) post-processing algorithm achieves better performance with faster inference speed than commonly used KNN. The experiment is conducted on the validation set of SemanticKITTI. The hardware used here is Nvidia RTX 2080 TI. It is also worth saying the CNN structure only needs 11ms to process one sample. Although we use the mix-precision choice provided by PyTorch during training, this inference speed is still remarkable. We credit this benefit to the parameter-free FID module.

TABLE II

PERFORMANCE COMPARISON BETWEEN PROPOSED NEAREST LABEL ASSIGNMENT AND COMMONLY USED KNN.

resolution	Network Modules	mIoU (%)	process time(ms)
64×2048	CNN +	55.4	11
	K-Nearest-Neighbor	58.7	2.7
	Nearest Label Assignment	**58.9**	**1.2**

V. CONCLUSION

In this paper, we discuss how to design the neural network solution for projection-based LiDAR point cloud segmentation. We propose a pipeline with an input module, a regular backbone, a FID module for upsampling, a classification head, and a post-processing step named NLA. In each module, we first try to choose the most common network settings, then argue why the chosen setting works.

Our pipeline achieves good performance on the benchmark and keeps a simple structure which we believe is hardware-friendly for both GPUs and onboard processors. We released our code, to make further development easier.

REFERENCES

[1] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 04:56:55 UTC from IEEE Xplore. Restrictions apply.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[5] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[6] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 108–11 117.

[7] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast semantic segmentation of lidar point clouds for autonomous driving," *arXiv preprint arXiv:2003.03653*, 2020.

[8] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. Liu, "2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network," *arXiv preprint arXiv:2102.04530*, 2021.

[9] F. Zhang, J. Fang, B. Wah, and P. Torr, "Deep fusionnet for point cloud semantic segmentation," in *ECCV*, vol. 2, 2020, p. 6.

[10] Y. A. Alnaggar, M. Afifi, K. Amer, and M. ElHelw, "Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1800–1809.

[11] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 475–12 485.

[12] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[13] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, "3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5432–5439, 2020.

[14] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4558–4567.

[15] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on χ-transformed points," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 828–838.

[16] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[17] D. Kochanov, F. K. Nejadasl, and O. Booij, "Kprnet: Improving projection-based lidar semantic segmentation," *arXiv preprint arXiv:2007.12668*, 2020.

[18] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.

[19] L. Han, T. Zheng, L. Xu, and L. Fang, "Occuseg: Occupancy-aware 3d instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2940–2949.

[20] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," *arXiv preprint arXiv:2011.10033*, 2020.

[21] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *European Conference on Computer Vision*. Springer, 2020, pp. 1–19.

[22] V. E. Liong, T. N. T. Nguyen, S. Widjaja, D. Sharma, and Z. J. Chong, "Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation," *arXiv preprint arXiv:2012.04934*, 2020.

[23] M. Gerdzhev, R. Razani, E. Taghavi, and B. Liu, "Tornado-net: multiview total variation semantic segmentation with diamond inception module," *arXiv preprint arXiv:2008.10544*, 2020.

[24] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong, "Deep neural network compression with single and multiple level quantization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[25] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 264–11 272.

[26] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[27] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[30] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2530–2539.

[31] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3887–3896.

[32] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5589–5598.

[33] R. A. Rosu, P. Schütt, J. Quenzel, and S. Behnke, "Latticenet: Fast point cloud segmentation using permutohedral lattices," *arXiv preprint arXiv:1912.05905*, 2019.

[34] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Bayesian spatial kernel smoothing for scalable dense semantic mapping," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 790–797, 2020.

[35] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4376–4382.

[36] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 926–932.

[37] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9601–9610.

[38] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *arXiv preprint arXiv:1805.07836*, 2018.

[39] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4413–4421.

[40] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on September 11,2026 at 04:56:55 UTC from IEEE Xplore. Restrictions apply.